AUSTRALIAN OS9 NEWSLETTER

sides 'No. of cylinders' (in decimal) :Interleave value: (in decimal) @FREE Syntax: Free [devname] Usage : Displays number of free sectors on a device @GFX Syntax: RUN GFX(<funct><args>) Usage : Graphics interface package for BASIC09 to do compatible VDG graphics commands @GFX2 Syntax: RUN GFX2([path]<funct><args>) Usage : Graphics interface package for BASIC09 to handle ... ax: none

Usage : ... graphics/
window ... on-line
help to ... help topics
@IDEN ... ormation
from OS ... -s = use
single lin ... execution
directory ... a device
@INKE ... outine to
input a ... abort to
the pro ... nk to a
memory ... ntents of

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
	Fax Messages	(07) 372-8325
SUPPORT	Brisbane OS9 Users Group	

text files @LOAD Syntax: Load <pathname> [...] Usage : Loads modules into memory @MAKDIR Syntax: Makdir <pathname> Usage : Creates a new directory file @MDIR Syntax: Mdir [e] Usage : Displays the present memory module directory Opts : e = print extended module directory @MERGE Syntax: Merge <path> ... ndard output @MFREE Synt ... RAM memory @MODPATCH ... a module in memory from ... warnings -c = compare modul ... odname = link to module C o ... te V = verify module M = ma ... Montype [opt] Usage : Set mo ... monitor m = monochrome m ... sage : Creates and links an OS ... ROCS Syntax: Procs [e] Usage ... em Opts : e = display all proc ... e : Prints the current data directory path @PXD Syntax: Pxd Usage : Prints the current execution directory path @RENAME Syntax: Rename <filename> <new filename> Usage : Gives the file or directory a new name @RUNB Syntax: Runb <i-code module> Usage : BASIC09 run time package @SETIME Syntax: Setime [yy/mm ...

... TPR
Syntax: ... to
num @ ... er

@TMODE Syntax: Tmode [pathname] [params] Usage : Displays or changes the operating parameters of the terminal @TUNEPORT Tuneport </t1 or /p> [value] Adjust the baud value for the serial port @UNLINK Syntax: Unlink <modname> Usage : Unlinks module(s) from memory @WCREATE Syntax:

OS9 ECHO

We are pleased to advise of a good deal of progress on the FIDO OS9 ECHO since our article last month. Access to the OS9 Community Network is now available in Australia and New Zealand through the FIDO worldwide network.

Now that the OS9 ECHO is being brought into Australia, Fido Zone 3, which, as I understand it, includes New Zealand, we have the potential for a very low cost means of staying in touch with the OS9 scene worldwide. There is a cost for this service which at the moment is being borne largely by ONE Fido Sysop in Brisbane on a trial basis.

I do not claim to be an expert on the Fido mail system but will try to explain the basics of the system as they will affect us. Email messages are posted to a local Fido BBS at the cost of a local telephone call which will be directed to the 'Echo' or to a specific Fido address. All messages are sent in a 'packet' to a HUB. These 'packets' are in simple terms transfered from one BBS to another in the Network and messages posted to EVERY BBS in the network (can be worldwide) or private messages end up at a single specified BBS which can be anywhere in the worldwide network.

The cost of bringing in the OS9-ECHO into zone 3 is estimated at $40.00 per month. So if only ONE user was to access this Echo, it would be reasonable to expect a charge of $40.00 per month. TWO users could share this cost at $20.00 per month, and so on. This reaches a point where a BBS Sysop would charge a very small fee or may provide this FREE. As in, no charges other than their normal membership fee. Typically, BBS membership would be appoximately $35 to $40 per year.

Now here is the crunch, we need as many Natgroup members as possible to share the cost of access to this OS9-ECHO used by the OS9 Community Network as a cost effective way of information transfer.

Fido BBS systems are available everywhere and there should be at least one in the local telephone district of each Natgroup member.

Log on to one of these and follow the membership application requirements of that BBS. Next, arrange with the Sysop for him to get the OS9-ECHO. The Sysop would be able to get details from Galaxy Gateway - Fido address

3:640/316 (James Collins is the Sysop of this BBS) telephone (BBS) 07-207-8900. This would immediately reduce the cost to the Galaxy Gateway as some of the cost is then borne by the Sysop in your area. This Fido network also offers a low cost means for US, the members of the National Usergroup, to stay in touch and share information.

I should point out that the Fido Network has some very definite rules which each user, and Sysop, is obliged to follow. Each BBS will list these rules; please take the trouble to read and understand them. As an 'OS9 Community' we do want to be seen as 'responsible' users. e.g. Private messages (question or answer to one individual) should be sent through FIDONET mail system as this avoids every Sysop on the network carrying, and paying for, the private message. Echo means that the message is 'echoed' to everybody on every BBS in the system that takes the OS9-ECHO.

MODEMS

If you do not have a Modem at present and are put off by the cost of a 2400 baud Modem, then maybe we can help. I used a 300 baud Modem for a number of years, and in fact, my used and perfectly good modem just sits on a shelf. If you would like an Avtek Mini-Modem II (300 and 1200/75), make an offer and I will send it. Rob Mackay also has one of these, plus an Avtek Multimodem (300, 1200/75 + 1200 Half Duplex) which he has offered for the price of a carton of beer plus postage. A Stop-press!! Don Berrie also has available a ACESAT 300, 1200/75 baud modem at about the same price.

Communication at 300 Baud is a little slow but it still costs no more than the price of a local telephone call. An RS-232 Pak could however be another problem. I see reports on the Echo of people running a Modem through the CoCo serial (bit-banger) port, so we plan to run some tests to see how reliable this is before suggesting this approach.

We also have a number of P.D. terminal programmes available; uploaded to Galaxy Gateway in Brisbane, you can get them from there, or send a disk and they are yours.

Well there we have it, we have arranged this worldwide access to information on our favourite operating system, so it is now over to each of you to support this important venture. If we all get behind it, the cost to

each of us will be very little. We have had a number of requests over the past four years for the Natgroup to run a BBS, now we can achieve something better and avoid STD charges.

Cheers, Gordon.

ooooooooooo0000000000oooooooooo

THE OS9 L2 UPGRADE STORY
By: Kevin Darling.

BACKGROUND Apr-Sep 88 - prehistory, private work
That April, Kent Meyers and myself got wind that an update was coming and we began passing along any bugs we found to the CoCo guys at Tandy. Partly because MW was getting out of the 6809 by then, I got the job.

At the end of September, MW flew me up to Des Moines to get info needed for two projects at the same time: the L-II update, and a port of something to the Atari ST. Both were supposed to be done by the end of that year; the plan was to first do all the known L-II bugfixes, and also add any enhancements possible within a few weeks. And that was all, at first.

In October, the ST project got cancelled. Otherwise, the L-II update would've been out way back then, and yet you'd have spent $20-30 for nothing (in comparison to what you've gotten free off the nets since).

Actually we'd be worse off than that, as most serious work would've stopped and there'd have been _no_ fast grfdrv, new gfx2, clock patches, acia speedups or a dozen other neat giveaways which came along... much, much later on.

NEW ORIGINAL SCHEDULE Oct-Dec 88 - new timeframe for official upgrade
Now we suddenly had some more time for L-II work (tho not much more), and both MW and Tandy gave carte blanche to add whatever we wanted. (Applause!) Knowing that this update would also become the new stock L-II release, I wanted a lot in there ⟨grin⟩, but needed some help doing so.

>From that arose together for the common good, modifying their own drivers to come into upgrade compliance.

To say that all this was unique in the history of OS's, would be an obvious understatement!

Now, altho no one at Tandy/MW ever said 'Let's stop here' (and I did so a few weeks too late), there _were_ times that a version could've been released:

END OF FIRST WORK Jan   89 - original clean up work done
Okay, we could've stopped in January 1989. But that still would've left tons of possible stuff untouched. By this time we were also writing demo programs that went beyond what anyone else had tried, and discovered (as others have recently) that Windint had major problems with multiple overlays and menus.

I should've collected the money and run right then, but ⟨sigh⟩ I wouldn't have been able to face myself later if I had done so. Dumb? Maybe. Hmm. But remember, this was going to be the new stock L-II...

GOING BEYOND Feb-Mar 89 - rewrite windint, numerous major and minor fixes   Apr-May 89 - grfdrv crunch and speedups applied, CLEAR problem solved

So we all continued, and I rewrote much of Windint. In the meantime, Kent was carving out huge chunks of room for me in Grfdrv, and that let me create those Grfdrv speedups you have now. We also were able to find a lot of new and really obscure/rare bugs, and fix them. Best of all, I also worked out a simple and effective fix for the irq stoppage when changing MW screens (and you already have that too! included in Bruce Isted's IBM mouse patch).

Thus the end of May 1989 was the next best stopping point, and in fact that's pretty much the version I myself still run today. The volunteer work was still ongoing tho... and something else happened: remember the RainbowFest when CerComp's 'Window Master - you don't have to be an OS-9 rocket scientist to have overlapping windows' RSDOS program came out? Kinda miffed me :-) 'HA!', I gritted my teeth, 'you wanna see easy-to-use overlapping windows?!'

GOING TOO FAR   Jun-Aug 89 - overlapping, moveable, resizeable windows, new gfx2

Sure :) From June to the end of August, 'super-windows' was written. After enormous work... 3-D look, moveable, resizeable, non-stopping, overlapping windows were crammed into the original 8K Grfdrv space. Still proud of that! You could even pick up windows and move them to other screens. Too neat.

Too neat. And useless. Great for demos and fun to play with (Dale Puckett loved 'em), but they took up slightly too much RAM, slowed down the covered windows, and compared to the full-screen apps we were used to: just

useless. Overlapping windows make a heckuva lot more sense on a larger display, yah? Also, they were impossible on a 128K system, which was part of the specs . So I think they'll remain a curiousity piece forever. Or probably go in OSK. Anyway, I decided to yank those and put back in other, more needed, stuff.

But y'all got much of the new GFX2 from that part of the project, too!

SUDDEN DEATH   Sep   89 - coco cancelled
Without any warning (which would've caused us to immediately wrap up), the CoCo got cancelled... just as everyone involved was 99% done (I'm sure even the remaining Tandy CoCo guys were caught totally off guard).

Naturally, that officially killed the upgrade... and the first Catch-22 situation arose: Tandy would consider selling it through EOS, if it was first submitted to them completely done, doc'd, and packaged ready-to-sell.

But no one had the money to do that. See, cancelled project = no payment :(And to do all that, without knowing for sure if it would be accepted? No way.

Plus, at exactly that moment, the new 68K machines were being born, which meant all our efforts had to turn in that future direction instead. Not long after, the CoCo EOS program was stopped. So the new upgrade got shelved. Well, not totally... bits and pieces have been posted the whole time since, and I think we all expect that sooner or later, it'll fully come out.

OTHER COMMENTS
We honed that sucker to a fare-thee-well! Absolutely the sweetest OS-9 version around... smooth, fast, 99.999% bugfree, crunched down code. Honestly, OSK sometimes feels clunky in comparison... I think others agree. (Note: we'd change some things nowadays, because the 128K limit is gone)

Again, you already have more stuff posted than the original spec called for. Anything after that has been gravy. Free gravy, too :-)

THE PEOPLE, AND WHAT WE PROVED
Even tho the upgrade got abruptly cancelled, I believe we easily proved that there's a _lot_ of talent out here that MW can and should make use of... not only by farming out paying projects, but even down to using volunteers to check over any of their current code (it's amazing what a third person can find and fix, or at least optimize).

[Incredibly, many of the volunteers are normally well paid professionals (at least two coders were people MW had failed to entice to Des Moines :-} I'd guesstimate at least .25 million dollars worth of time was _donated_ willingly, just to help out the project. I _love_ these guys!!]

Another good point, btw: who better to do an update, than the people who have to write programs and drivers for and/or use that system daily for personal use?

I must also note the unbelievable secrecy, which was held for well over a year (until Tandy said it was okay to talk) by everyone. Sure, all involved signed nondisclosure agreements, but still it was unprecedented. I didn't even see secrets held that well when I did NSA-related work.

Another thing that was proved beyond a shadow of a doubt: telecommuting is viable! You couldn't find a more scattered group of people on this continent, yet the project progressed smoothly and in coordination (in some part due to the CIS forum space and lots of phone calls :-}. The most astonishing thing was that even while they were being independently innovative, not one person ever balked at my telling them that a certain method was unsuitable... instead they just used such rules as a baseline to do even more superlative work.

So whether the "full" upgrade comes out or not, all the volunteers deserve the utmost praise and high marks for their work.

Anyway, now you know... (most of) the rest of the story :-) regards - kevin.

ooooooooooo00000000000ooooooooooo

FIRST IMPRESSIONS OF THE MM/1

A few people reading this will know that I ordered an MM/1 way back on the 26th of March, 1991. I ordered the complete MM/1 Extended Kit. At the time it was selling for US$875, which I paid, it is now US$975.

The MM/1 Extended is the MM/1 mother board and the accompanying daughter board or I/O board. Complete with software and manuals and a disk drive.

I rang Paul Ward of IMS in Washington a couple of times after that, just checking what was (or wasn't!) happening. Finally on the 20th of December last year I got a lettergram from DHL in Sydney!! Yep, it was here! It only took another week and a half to get it to Melbourne, out of bond, and into my hands. It seems the Compass crash also had a little to do with this delay!

I was charged $260 import duty and the original US$975 ($875 + $100 shipping) became $1275 Australian. For a little over $1500 I think I've got a bargain!

Some background! The MM/1 is an OS/9 68000 based computer. This is often called OS9-68K (K=1000) or OSK. OSK is the big brother of OS/9, made for the bigger, more modern 68000 family of processors. OS/9 for the 6809 is no longer sold or supported by Microware.

The MM/1 uses a Signetics 68070 processor. This thing runs at a little under 16MHz. It is designed for multimedia applications, for example CD-TV. It has built into it such things as serial ports, timers and data channels. The other main chip in the MM/1 is the VSC or Video and System Controller. This chip does all the graphics and memory management. Sort of like a cross between the CoCo's GIME chip and the Amigas blitter!

(See attached article about the 68070 and VSC chips .. Ed)

The main board contains 1 Meg of memory, has two serial ports and all the necessary stuff like video and keyboard plugs. The I/O board (which I'm still waiting for) has the SCSI hard drive controller, sockets for more memory, stereo sound input and outputs, 3 more serial ports and 2 parallel ports.

I've mounted the main board in a mini tower case (normal IBM type), I'm using a normal XT type keyboard, and until recently a CM8 monitor. I also hard wired the megahertz lights on the case to "68".. just to upset IBM types!

I've also got my two 1.44 meg 3.5" drives, plus a 1.2 meg 5.25" drive in there as well.

The MM/1 main board is 4"x8"! Thats smaller than a floppy drive!
I haven't recieved my I/O board yet, or my microware OSK manuals, but I did get the IMS manuals and 6 high density disks full of software.

The disks contained all the OS9 stuff needed like commands, and modules directories. Also an assembler, basic (called microware basic- its the same as basic09) and a C compiler. Heaps of graphics and sound demos, some communications programs and lots of sample programs.

I've also filled heaps of disks with PD software I've gotten through the Internet. A friend from Holland sent me 12 disks of demos and programs and I'm grabbing GIF pictures pretty quick too.

The MM/1 comes with a program to display autodesk animator (.fli) files. These are 256 colour fully animated files. Some are extremely good! I have one of

a piece of patio furniture flying onto the screen, unfolding, then flying around the grand canyon! The speed of the MM/1 really shows with things like this.

Also there is a kind of famous mouse demo of a mouse getting a fright and running off the screen. This one is so fast it really needs to be seen!

The MM/1 does not have text screens. It uses graphics screens and draws the letters using whatever font you like. This is very similar to the CoCo's graphics windows, but due to a part of the VSC chip, called a Pixel Accelerator (PIXAC), this is done mostly without the help of the main processor. The text scrolls past faster than the CoCo's 80 column text screens.

Fonts can be taken from CoCo OS9 Level II and used on the MM/1 by just changing one byte.

At the moment, the MM/1's windows look much like the CoCo's. You can flip through them, create overlays and all the normal stuff. Eventually it is meant to have moveable, resizeable windows. That will be nice to see. (This stuff is supposed to be on the IMS update disks.. still waiting for them too!)

Apart from looking good in the tower case, I've found that with no hard drive, at least two drives is a really good idea. With 1Meg of RAM, I get about 750k free after booting. Thats not enough to use a randisk and run the big animation files, but its plenty to stick C source in the ram disk and use only one floppy.

The C compiler is outwardly quite similar to the CoCo's. Except that it uses sticky modules. Sticky modules are modules which stay in memory until that memory is actually needed. Any executable program can be a sticky module. When the MM/1 is really moving, it has all the C compiler modules in memory already, and compiles small programs in about one second!

All of OSK is much more detailed than CoCo OS9. For example, even the dir command has almost a page of options!

So far I've only needed to make one new boot disk! (Something most people hate doing with the CoCo). This was made so I can use my third disk drive and also to install a newer 'windio' module. (windio does all the windowing stuff.. this module is being updated by IMS).

The disks contained documentation to almost everything on them. IMS sent me three real manuals. A hardware one which explains how to put it all together. A software one which explains simple commands and how to use the application programs they sent me. And a manual for Oddjob which is a simple language also included with the

MM/1, much like awk for unix.

The official Microware manuals are still on their way to me. They contain all the commands and technical info. Also the Basic and C manuals are in there.

When I set up my MM/1, I moved all the plugs onto the back of my tower case. The video and /t0 ports both use DB9 connectors, so I needed two of these, and the keyboard plug uses the normal five pin din. IMS provides a disk drive cable.

Power is supplied to the main board from the two standard IBM power supply cables, which plug into the mini-back

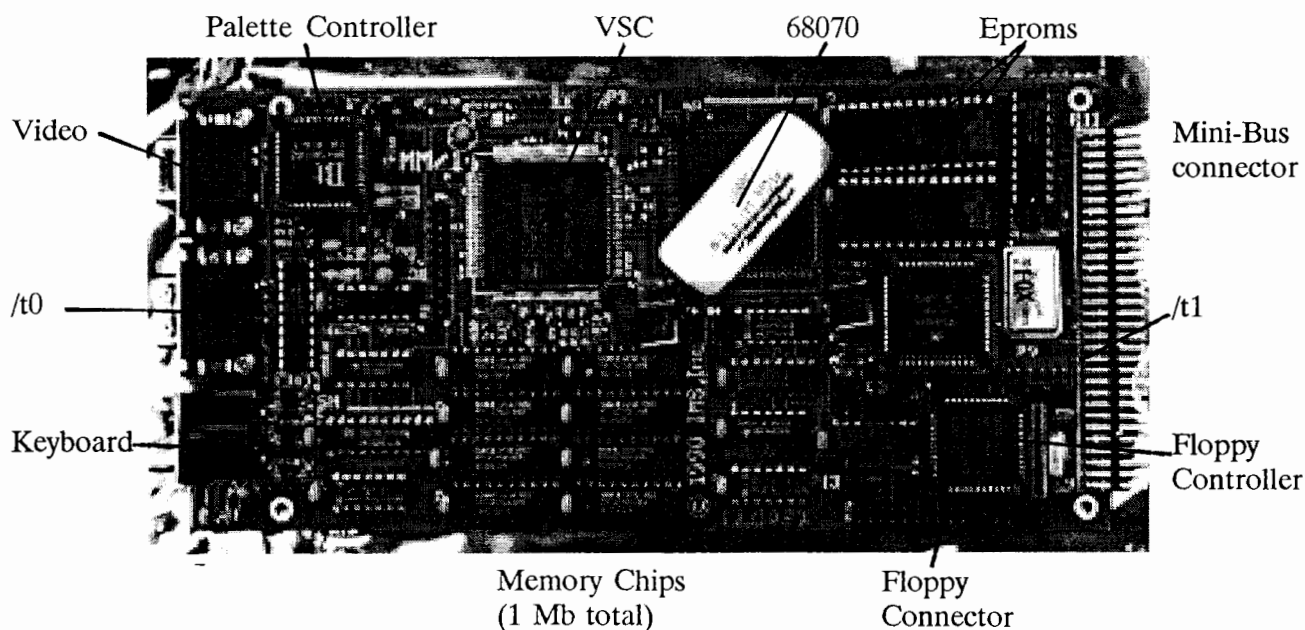plane. The main board is then plugged into that.

Some more info...

MM/1's should be available in Australia soon, from a company called Multi-Media Australia. This is run by a guy called Jerome Slappy. I havn't contacted him yet. This info is from Paul Ward.

IMS has a BBS set up in the United States to give users upgrades of their software faster. I still havn't gotten around to ringing this. I'd rather put it off!

Hope this information helps - Andrew Donaldson.

oooooooooo0000000000oooooooooo

MM/1 Main Board

Palette Controller — VSC — 68070 — Eproms — Video — /t0 — Keyboard — Mini-Bus connector — /t1 — Floppy Controller — Memory Chips (1 Mb total) — Floppy Connector

68070/VSC Specifications and Features

I hope I'm not rocking any boats, but alot of people are curious about 68070/VSC specs, and nobody "in the know" has revealed much. So I'm gonna blow the doors off the SIG and reveal all. Note: while I think every fact in this file is correct, I wouldn't bet $1000 on it!

I place this file totally in the public domain... copy away!

Matthew Thompson
Ottawa, Ontario, Canada

The SCC68070 features:

- Made by Signetics (Philips), uses CMOS tecnology (low 45mA current drain).

- 99.9999% MC68000 software compatability.

Exceptions:

- Implements full bus recovery (like the MC68010), adding a measure of crash-proofing to the system. However it only stacks 17 words during recovery, instead of the 68010's 29 (which the 070 manual claims is redundant, anyways). - In the System State the 68070 stacks four words (like the 68010), instead of the 68000's three. - Other than that, they are identical.

- Runs at 15 MHz, about twice as fast as standard 68000-based Amigas, Macs, and Ataris.

- Cycle times for 68070 instructions are different than on a 68000, though. Most instuctions take more. But at 15 MHz, they're all faster than any of the above 68000 systems (heh heh). A Bus Cycle is 4 Clock Cycles, while a Machine Cycle is 3 Clock Cycles. Most of the handiest instructions have been cranked somewhat faster, such as MOVE.

- Two high speed DMA channels, each capable of 2.5 Mega-xfer/sec I/O from/to RAM xfers (one channel at a time, of course). Channel 1 has a higher priority than channel 2, and channel 2 can also do 1.25 Mega-xfer/sec RAM to RAM block copies. Xfers can either be in bytes or words. Supports daisy chaining to external DMA chips for more channels. The DMA ports on the 68070 are identical in register structure and function as the Motorola MC68430/40/50 DMA chips. All arbitration is handled by the 68070. *:Equivalent to copying thirty-nine 320 x 200 x 256-color screens per sec!

- I`2C bus interface (I`2C is an industry standard - 100kbaud serial interface intended for el-cheapo chip interfaces. Instead of complex decoding chips and parallel bus wiring, any I`2C device (such as configuration EPROMS, time clock ICs, serial port chips, joystick control chips, etc., etc.) can all be hooked together by a couple common wires, and then everything is handled & figured out by on-chip arbitration circuitry. Major reductions in wiring and design complexity are achieved.).

- On-chip serial interface. Features seperate baud rates for receive and xmit; can generate interrupt at any priority level; Odd/Even/No parity; 7/8 bits; built in CTS and RTS lines; full/half duplex operation; framing error detection; and more. Using an external clock of 4.9152MHz, you can get baud rates of 75, 150, 300, 1200, 2400, 4800, 9600 and 19200. Using a clock of 8MHz you can also get 31,250 baud (MIDI speed).

- Also has built in MMU, with 8 descriptors on-chip, up to 128 with external bipolar RAM. Supports intermodule

protection; stack overflow protection; and also controls read/write/execute/supervisor access to any segment of memory. OS/K wisely does not use the MMU, however, since like most MMUs on high-speed CPUs it throws in an extra wait-state, slowing the CPU by as much as 20 percent.

- Three 16-bit timers on-chip. All have a programmable increment time down to a minimum of 6.51uS. One is a free-running counter with auto-reload. Two are capture counters, which are controlled by seperate I/O pins on the chip. They can generate pulses, or count the time between input transistions, or even count the number of input transistions. All timers can generate an interrupt to the CPU.

- Built-in prefetch cache speeds things up a bit.

- All this on just one 84-pin, square, surface-mount flat-pack chip.

- Clearly designed with one single purpose in mind: to cram as much power on one chip at the lowest cost. Hence its use in our low-cost, but high performance, MM/1 and TC070 computers.

The VSC chip features:

- Real part number (hidden until now from Delphi members) is the SCC66470, (not 68470) and is a 124-pin flat-pack CMOS chip from Signetics (Philips).

- Runs at up to 30MHz.

- Max 768 by 560 resolution.

- Max 256 colors at 384 by 560, 16 at 768 by 560, 256 at 768 by 560 if a second 66470 is connected in more-or-less parallel. I am not sure if such a piggybacking scheme will be officially supported for the MM/1 or TC070 (this is not the same as the palette board being offered).

- Supports 31.5 kHz horizontal scan if your monitor can crank it.

- Has on-chip DRAM and SRAM controller which supports bit, nybble, page and dual port RAMs.

- Max 1 Meg video RAM, extra 0.5 Meg DRAM and 0.5 Meg SRAM supported.

- Built in Blitter (PIXAC - PIXel ACcelerator).

- Has all kinds of blitting features that make block and pixel blitting fast and easy. Each blitting operation affects a word value (ie up to 2 or 4 pixels depending on mode) at a time, and each word "takes less than 500nS to process." For instance, the 66470 can take a one-pixel-

per-bit image definition (such as a font), convert those bits into nybbles or bytes of the correct foreground, and blit the result on the screen while masking the background using a transparent color, so that only those bits that were set affect the screen. On top off all this a programmable logic operation between the source and destination pixels can be thrown in for good measure. A 64 x 64 screen blit on a 16-color screen would only take (a guesstimated) 512uS or less. About the only way to blit faster is if all you want to do is a plain vanilla copy, thus the DMA could copy a 64 x 64 16-color block in about 45 uS. Anything else would require a fancy CPU algorithm that would take mega time. And while the VSC is blitting, the CPU can do other things. The VSC blitter can also allign source nybbles to destination nybbles, so that any pixel boundary can be used.

- Block blit functions include COPY (raw), PATCH (raw with xparent masking), EXCHANGE (switch two image areas), SWAP (switch with xparent masking), COLOR1 (take all non-xparent source pixels and turn destination into foreground color), COLOR2 (like COLOR1 except xparent pixels -> background), BCOLOR1 (like COLOR1, except performs bit-to-nybble or bit-to-byte expansion), BCOLOR2 (combo of COLOR2 and BCOLOR1), COMPARE (compare pixels, set flag if match), and COMPACT (compare pixels, create a table of flags).

- The reccommended color scene is RGBI for 16-color screens, and RRRGGGBB for 256-color screens (blue is less noticeable hence it is allotted two bits of resulution).

- Also supports smooth scrolls and virtual windowing.

- Can also shrink or zoom by a factor of two independently in the horizontal and vertical directions during all blits. Instant multisize fonts!

- The Amiga's blitter is a "monochrome" blitter. To affect a mutlicolor screen it has to blit each Amiga bit plane seperately. The VSC uses a nybble/byte scheme and does all color computation in one access. The Amiga's blitter is also so complicated to program that the current release of the operating system uses the CPU to blit text, because it takes so long to set up the blitter that it would actually take longer. The VSC was made for blitting text, and for all kinds of block copies and exchanges. The Amiga's blitter may still be faster at one-pixel-wide operations, like random-angle diagonal lines, or curved lines. I won't know till I see the bottom lines for the two machines.

- Pixel and block testing, making collision detection in games and etc. incredibly fast.

- On average 3 to 8 times as fast as a 68000 alone, even more.
- Can cut algorithm sizes by up to 98%.

- Genlock and frame grabbing support with some extra logic and a PLL, and for frame grabbing, a flash-ADC convertor. Real-time, 60 frame-per-second image digitization is easily possible.

- Based on the specs for this baby, we ought to see some impressive graphics games, like Microsoft Flightsim V4.0 out running on a 320 by 200 by 16-color screen at a frame rate of, oh, say... 10 per second!?

- Also has built in reset ciruitry, eliminating more discrete parts.

- Support for watchdog timer.

- Designed with one purpose in mind: to crank the graphics speed to the max at the lowest cost. And since it is designed to work hand-in-hand with the 68070, it's the obvious choice for a gfx chip for the TC070 and MM/1.

Note that the MM/1 and TC070 only go to 480 vertical resolution max because to get 560 you have to use a 625-line 50Hz-scan standard like PAL (but the flicker would be bad). NTSC is 525 lines at 60Hz. The max horizontal is 720 for similar reasons.

Also, note that normally the 66470 arbitrates between itself and the 68070 as to whose turn it is to look at RAM. It does this in the single-board version of the MM/1. But both the full MM/1 and TC070, which support over 2 Meg (way beyond the 66470 DRAM controller), obviously have seperate DRAM controllers and semi-seperate buses. The VSC gets a Meg to itself, the CPU, gets the rest to itself, and they both crank along at maximum, zero-wait-state speed. The 68070 doesn't have to wait for the VSC to grab a video word, and the VSC no longer has to wait for the CPU to be finished to get or put a blitting word. So they BOTH speed up (despite some kidding on the forum!). The 68070 can still access the video RAM, of course, but it might have to wait (a VSC bus access is 4 clock cycles, the same as the CPU).

Put it this way: With these two chips at the heart of the MM/1 and TC070, it will quite some time before the competition gets as much speed and features for as low a cost as either system! --

Brian White (Using a friends account)

ooooooooooo0000000000ooooooooooo

A C Tutorial
Chapter 6 - Defines and Macros

DEFINES AND MACROS ARE AIDS TO CLEAR PROGRAMMING

Load and display the file named DEFINE.C for your first look at some defines and macros. Notice the first four lines of the program each starting with the word "#define". This is the way all defines and macros are defined. Before the actual compilation starts, the compiler goes through a preprocessor pass to resolve all of the defines. In the present case, it will find every place in the program where the combination "START" is found and it will simply replace it with the 0 since that is the definition. The compiler itself will never see the word "START", so as far as the compiler is concerned, the zeros were always there.

It should be clear to you by now that putting the word "START" in your program instead of the numeral 0 is only a convenience to you and actually acts like a comment since the word "START" helps you to understand what the zero is used for. In the case of a very small program, such as that before you, it doesn't really matter what you use. If, however, you had a 2000 line program before you with 27 references to the START, it would be a completely different matter. If you wanted to change all of the STARTs in the program to a new number, it would be simple to change the one #define, but difficult to find and change all of the references to it manually, and possibly disastrous if you missed one or two of the references.

In the same manner, the preprocessor will find all occurrences of the word "ENDING" and change them to 9, then the compiler will operate on the changed file with no knowledge that "ENDING" ever existed. It is a fairly common practice in C programming to use all capital letters for a symbolic constant such as "START" and "ENDING" and use all lower case letters for variable names. You can use any method you choose since it is mostly a matter of personal taste.

IS THIS REALLY USEFUL?

When we get to the chapters discussing input and output, we will need an indicator to tell us when we reach the end-of-file of an input file. Since different compilers use different numerical values for this, although most use either a zero or a minus 1, we will write the program with a "define" to define the EOF used by our particular compiler. If at some later date, we change to a new compiler, it is a simple matter to change this one "define" to fix the entire program. End-of-line is another indicator that is not universal. This will make more sense when we get to the chapters on input and output.

WHAT IS A MACRO?

A macro is nothing more than another define, but since it is capable of at least appearing to perform some logical decisions or some math functions, it has a unique name. Consider the third line of the program on your screen for an example of a macro. In this case, anytime the preprocessor finds the word "MAX" followed by a group in parentheses, it expects to find two terms in the parentheses and will do a replacement of the terms into the second definition. Thus the first term will replace every "A" in the second definition and the second term will replace every "B" in the second definition.

When line 12 of the program is reached, 'index' will be substituted for every "A", and 'count' will be substituted for every "B". Remembering the cryptic construct we studied a couple of chapters ago will reveal that "mx" will receive the maximum value of 'index' or 'count'. In like manner, the "MIN" macro will result in 'mn' receiving the minimum value of 'index' or 'count'. The results are then printed out. There are a lot of seemingly extra parentheses in the macro definition but they are not extra, they are essential. We will discuss the extra parentheses in our next program.

Compile and run DEFINE.C.

LETS LOOK AT A WRONG MACRO

Load the file named MACRO.C and display it on your screen for a better look at a macro and its use. The first line defines a macro named "WRONG" that appears to get the cube of "A", and indeed it does in some cases, but it fails miserably in others. The second macro named "CUBE" actually does get the cube in all cases. Consider the program itself where the CUBE of i+offset is calculated. If i is 1, which it -is the first time through, then we will be looking for the cube of 1+5 = 6, which will result in 216. When using "CUBE", we group the values like this, (1+5)*(1+5)*(1+5) = 6*6*6 = 216.

However, when we use WRONG, we group them as 1+5*1+5*1+5 = 1+5+5+5 = 16 which is a wrong answer. The parentheses are therefore required to properly group the variables together. It should be clear to you that either "CUBE" or "WRONG" would arrive at a correct answer for a single term replacement such as we did in the last program. The correct values of the cube and the square of the numbers are printed out as well as the wrong values for your inspection. The remainder of the program is simple and will be left to your inspection and understanding.

PROGRAMMING EXERCISE 1.

Write a program to count from 7 to -5 by counting down. Use #define statements to define the limits.

(Hint, you will need to use a decrementing variable in   the third part of the "for" loop control).

ooooooooooo0000000000oooooooooo

An Index of Rainbow OS9 Articles
compiled by Bob Devries
January - December '88

January 1988 page 156
Stalking the Fire-Breathing Dragon - Tips for the OS9 beginner.
Nancy Ewart

January 1988 page 166
Catch the Wave - Help for OS9 users.
Cray Augsburg

January 1988 page 160
KISSable OS9 - Back at the drawing board.
Dale L. Puckett

January 1988 page 176
OS9 Programming - Screen dumping revisited.
Peter Dibble

February 1988 page 152
The Impact of Multi-Vue - A first look at Tandy's user-friendly interface for OS9 Level II.
Cray Augsburg

February 1988 page 182
KISSable OS9 - Using a fourth-generation database language.
Dale L. Puckett

March 1988 page 180
KISSable OS9 - A view of Multi-Vue.
Dale L. Puckett

April 1988 page 160
KISSable OS9 - New tools, new toys.
Dale L. Puckett

May 1988 page 178
KISSable OS9 - Patches, programs and politics.
Dale L. Puckett

June 1988 page 14
Help is on the way - Create online assistance.
Stephen B. Goldberg

June 1988 page 180
KISSable OS9 - Another great beginning.
Dale L. Puckett

July 1988 page 16
A new outlook for OS9 - Using subdirectories and shell scripts to build an OS9 menu system.
Mark Roseman

July 1988 page 174
KISSable OS9 - Sending the right signals.
Dale L. Puckett

August 1988 page 182
KISSable OS9 - Volunteers build a better mousetrap.
Dale L. Puckett

October 1988 page 147
KISSable OS9 - Another cry for standards.
Dale L. Puckett

November 1988 page 176
KISSable OS9 - Installation, automation and more.
Dale L. Puckett

December 1988 page 160
Parameter Changes Made Easy - Take the drudgery out of changing parameters.
Steve Goldberg

December 1988 page 178
KISSable OS9 - Better tools are here.
Dale L. Puckett

ooooooooooo0000000000oooooooooo

CoCo-Link
CoCo-Link is an excellent magazine to help you with the RSDOS side of the Colour Computer. It is a bi-monthly magazine published by Mr. Robbie Dalzell. Send your subscriptions to:
CoCo-Link
31 Redlands Crescent
Pt. Noarlunga Sth.
South Australia
Phone: (08) 3861647